TOSHIBA

Leading Innovation

Deadline Miss Detection with SCHED_DEADLINE

Yoshitake Kobayashi
Advanced Software Technology Group
Corporate Software Engineering Center
TOSHIBA CORPORATION

Resources

- **The latest version slide is available on elinux wiki**
  - http://elinux.org/ELC_2013_Presentations
- **Related source code is able to download from GitHub**
  - https://github.com/ystk/sched-deadline/tree/dlmiss-detection-dev

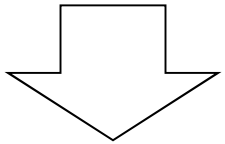
Outline

- **Motivation**
- **Deadline scheduler**
- **SCHED_DEADLINE and its evaluation**
- **Budget management**
- ***Deadline miss detection***
- **Conclusion**

Motivation

- **We would like to use Linux on control systems**
- **Real-time is one of the most critical topic**

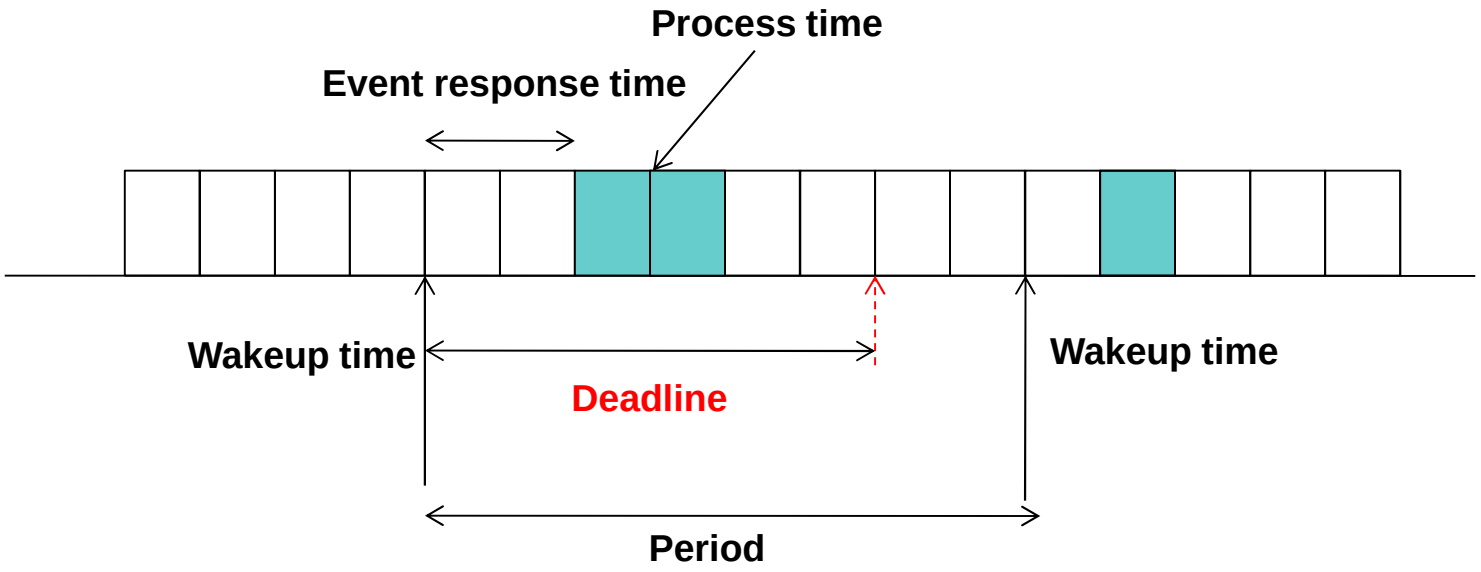
- **Problem statement**
  - Need to evaluate deeply to meet the deadline
  - CPU resource used too much by higher priority tasks



EDF scheduler

Definition of deadline

- **Wakeup time: The time of an event occurred (Ex. Timer interrupt) and target task's state changed to runnable.**
- **Event response time: Interrupt latency**
- **Deadline: The time for a target task must finish**



Earliest Deadline Fisrt scheduling（EDF）

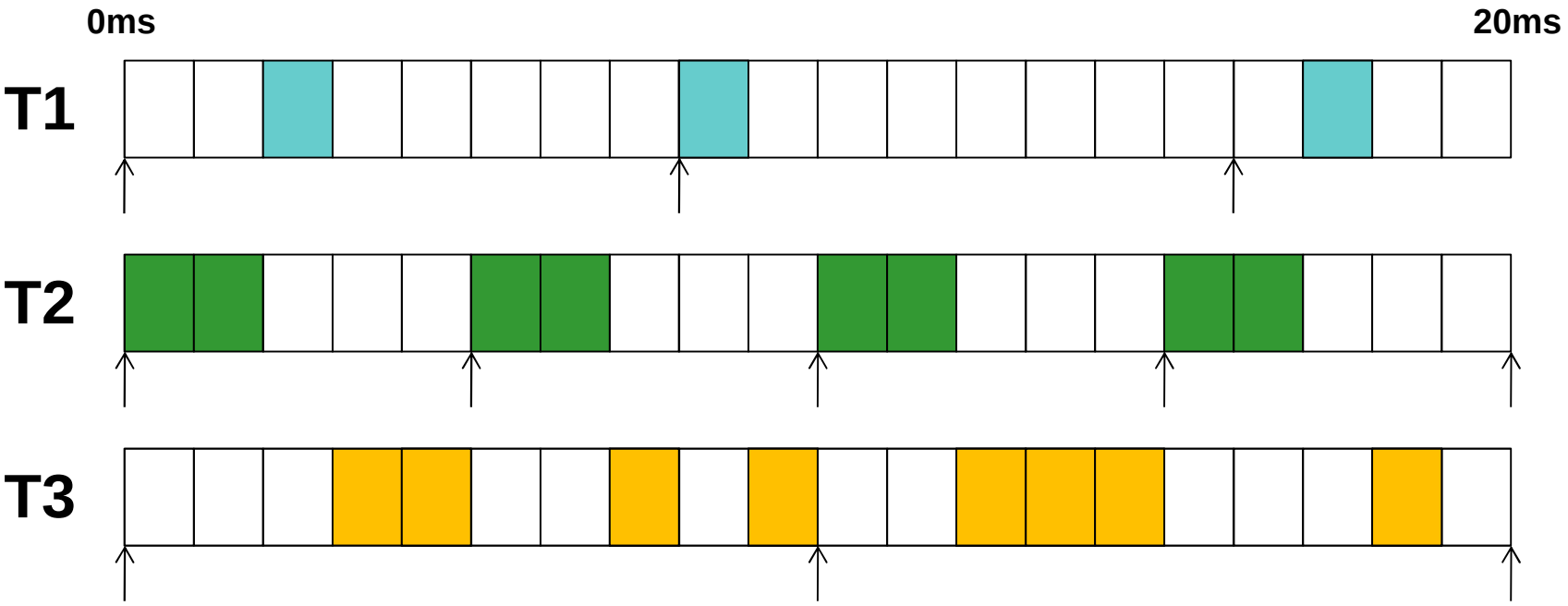
- **The earliest deadline task has the highest priority**
- **Task's priority is dynamically changed and managed**
  - SCHED_FIFO is static priority management
- **Theoretically the total CPU usage by all tasks is up to 100%**
  - Includes the kernel overhead
  - If usage of CPU by all tasks is less than 100%, all tasks meet the deadline

- **Reference**
  - http://en.wikipedia.org/wiki/Earliest_deadline_first_scheduling

An example of EDF Scheduling

- **Task1: budget 1ms　period 8ms**
- **Task2: budget 2ms　period 5ms**
- **Task3: budget 4ms　period 10ms**

CPU usage＝0.925%
＜ 100%



Rate-Monotonic Scheduling（RMS）

- **One of the popular scheduling algorithm for RTOS**
- **Assumptions for task behavior**
  - NO resource sharing such as hardware, a queue, or any kind of semaphore
  - Deterministic deadlines are exactly equal to periods
  - Static priorities (the task with the highest static priority that is runnable immediately preempts all other tasks)
  - Static priorities assigned according to the rate monotonic conventions (tasks with shorter periods/deadlines are given higher priorities)
  - Context switch times and other thread operations are free and have no impact on the model
- **CPU utilization**
  - n: number of periodic tasks, Ti: Release period, Ci: Computation time

$$U = \sum_{i=0}^{n} C_i / T_i \leq n(\sqrt[n]{2} - 1) \xrightarrow[n=\infty]{} \ln 2 \approx 0.69$$

  - CPU utilization depends on the combination of periodic tasks and it is possible to meet the deadline even the CPU utilization is around 80%

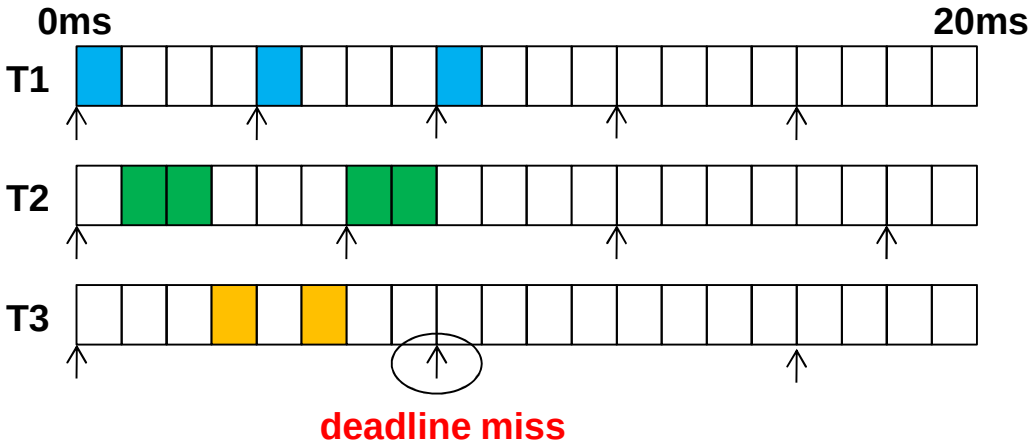
- **Reference**
  - http://en.wikipedia.org/wiki/Earliest_deadline_first_scheduling

Compared with the RMS scheduling

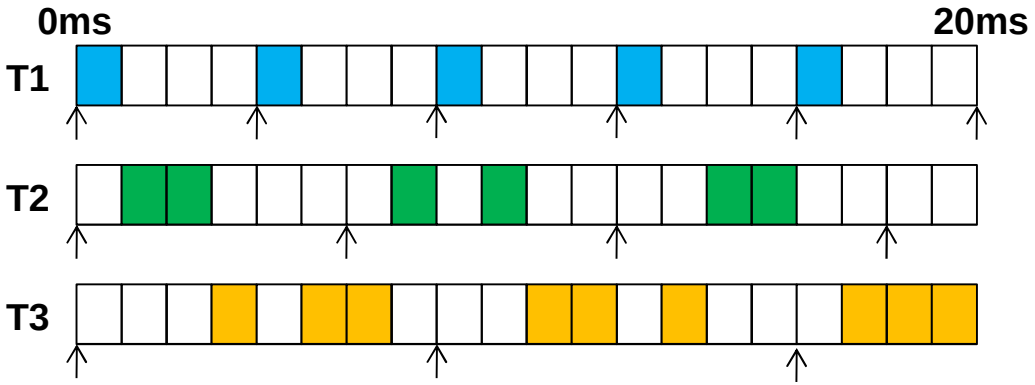
- **Task1: budget 1ms period 4ms**
- **Task2: budget 2ms period 6ms**
- **Task3: budget 3ms period 8ms**

CPU usage＝0.958%

RMS



EDF



Comparison of deadline algorithms

	Advantage	Disadvantage
RMS	Easier to implement	Evaluation for scheduling possibility is required to meet the deadline
EDF	No evaluation for scheduling possibility is required to meet the deadline	Difficult to implement

What SCHED_DEADLINE is?

- **http://www.evidence.eu.com/sched_deadline.html**
  - Implements the EDF scheduling algorithm
  - Posted to LKML by Dario Faggioli and Juri Lelli
  - Latest version is V7 (2013/2/12)
    - Our work based on V2 and V4.

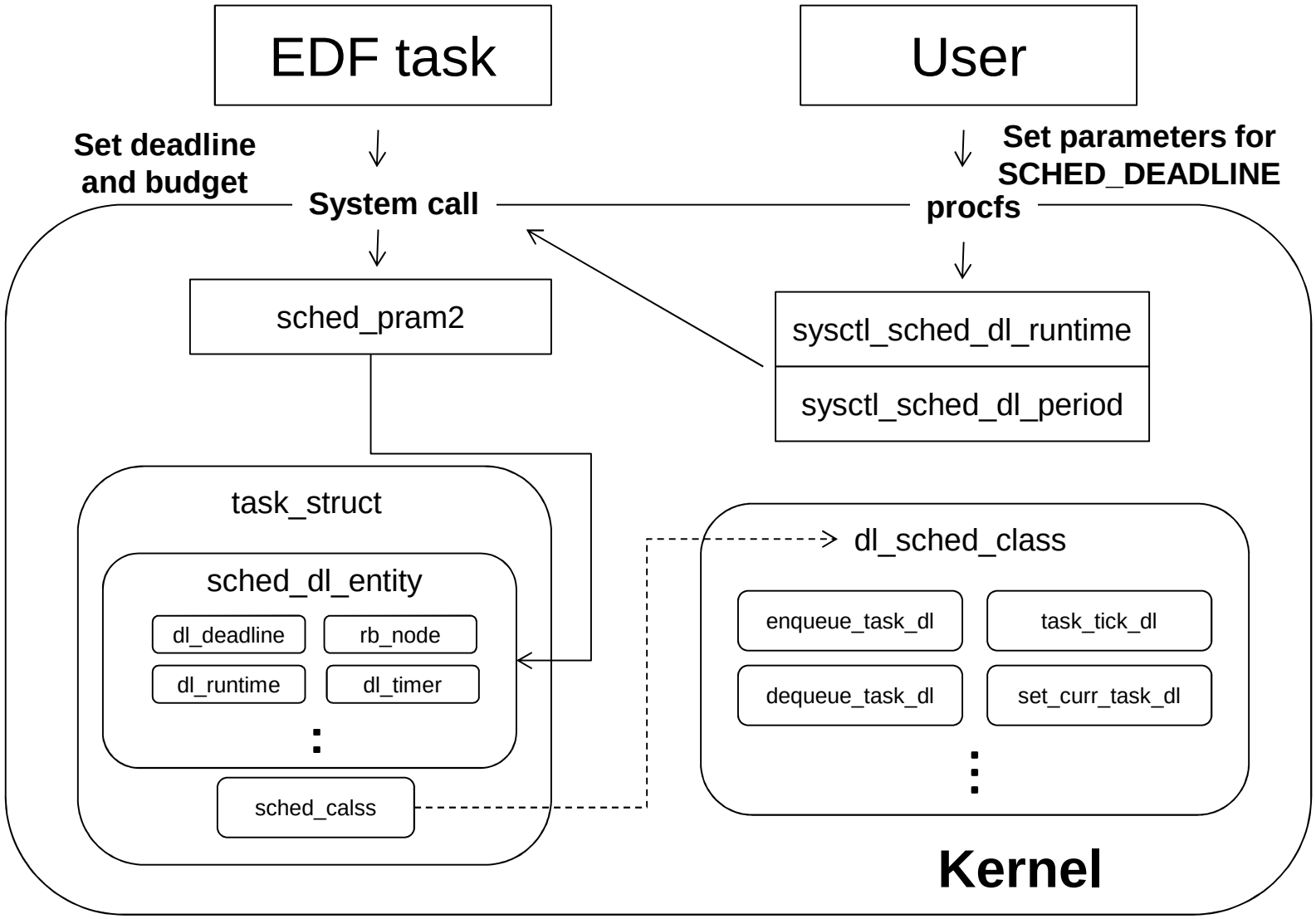
- **Key features of SCHED_DEADLINE**
  - Temporal isolation
    - The temporal behavior of each task (i.e., its ability to meet its deadlines) is not affected by the behavior of any other task in the system
    - Each task is characterized by the following aspects:
      - Budget: sched_runtime
      - Period: sched_period, equal to its deadline

Build SCHED_DEADLINE enabled kernel

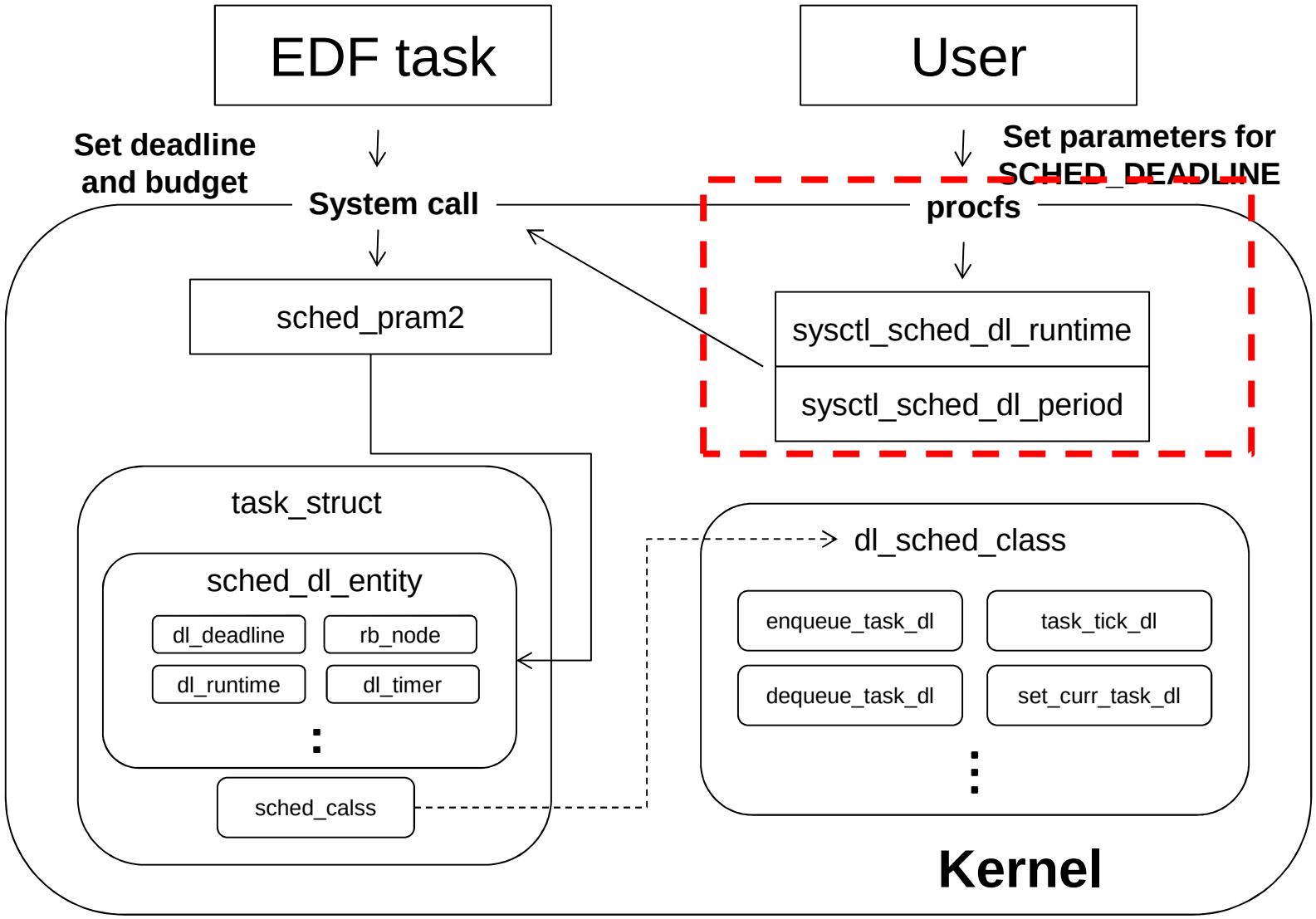
- **Get the source code from the following places**
  - git clone git://github.com/jlelli/sched-deadline.git (for V4 to V7)
  - git clone git://gitorious.org/rt-deadline (for V2)

- **Kernel configuration**
  - CONFIG_EXPERIMENTAL = y
  - CONFIG_CGROUPS = y
  - CONFIG_CGROUP_SCHED = n
  - CONFIG_HIGH_RES_TIMERS = y
  - CONFIG_PREEMPT = y
  - CONFIG_PREEMPT_RT = y
  - CONFIG_HZ_1000 = y

Overview of SCHED_DEADLINE



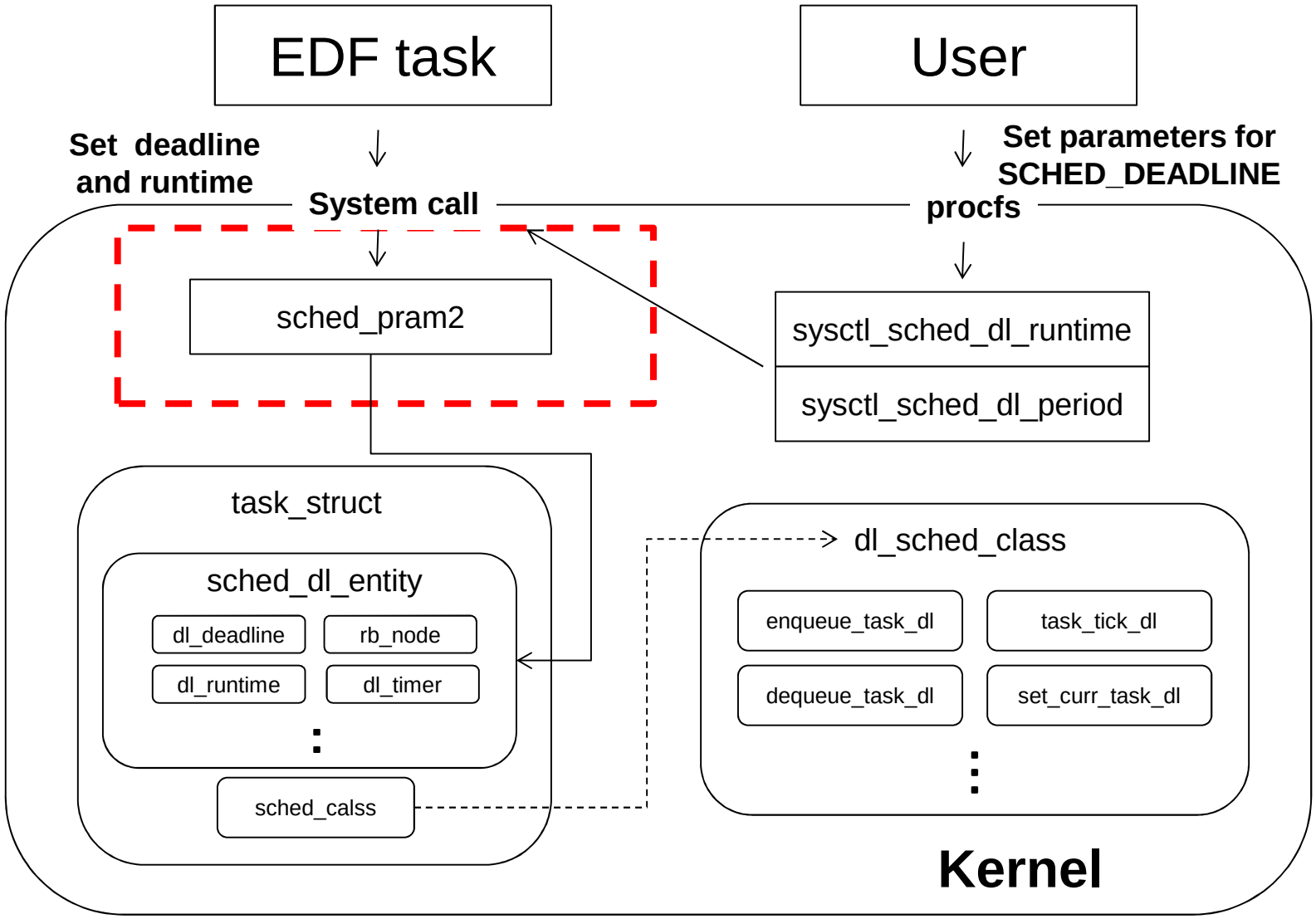
Overview of SCHED_DEADLINE



Setting CPU utilization for EDF tasks

- **Parameters can be setted via procfs**
  - CPU utilization for rt(SCHED_FIFO and SCHED_RR) and dl(SCHED_DEADLINE) should be under 100%
  - Parameters for EDF scheduler
    - /proc/sys/kernel/sched_dl_period_us
    - /proc/sys/kernel/sched_dl_runtime_us
- **When a task requires more than above limit, the task cannot submit to run**
- **An example setting （rt: 50%, dl:50%）**
  - # echo 500000 > /proc/sys/kernel/sched_rt_runtime_us (500ms)
    # echo 100000 > /proc/sys/kernel/sched_dl_period_us (100ms)
    # echo 50000 > /proc/sys/kernel/sched_dl_runtime_us (50ms)

Overview of SCHED_DEADLINE



Structure of sched_param2

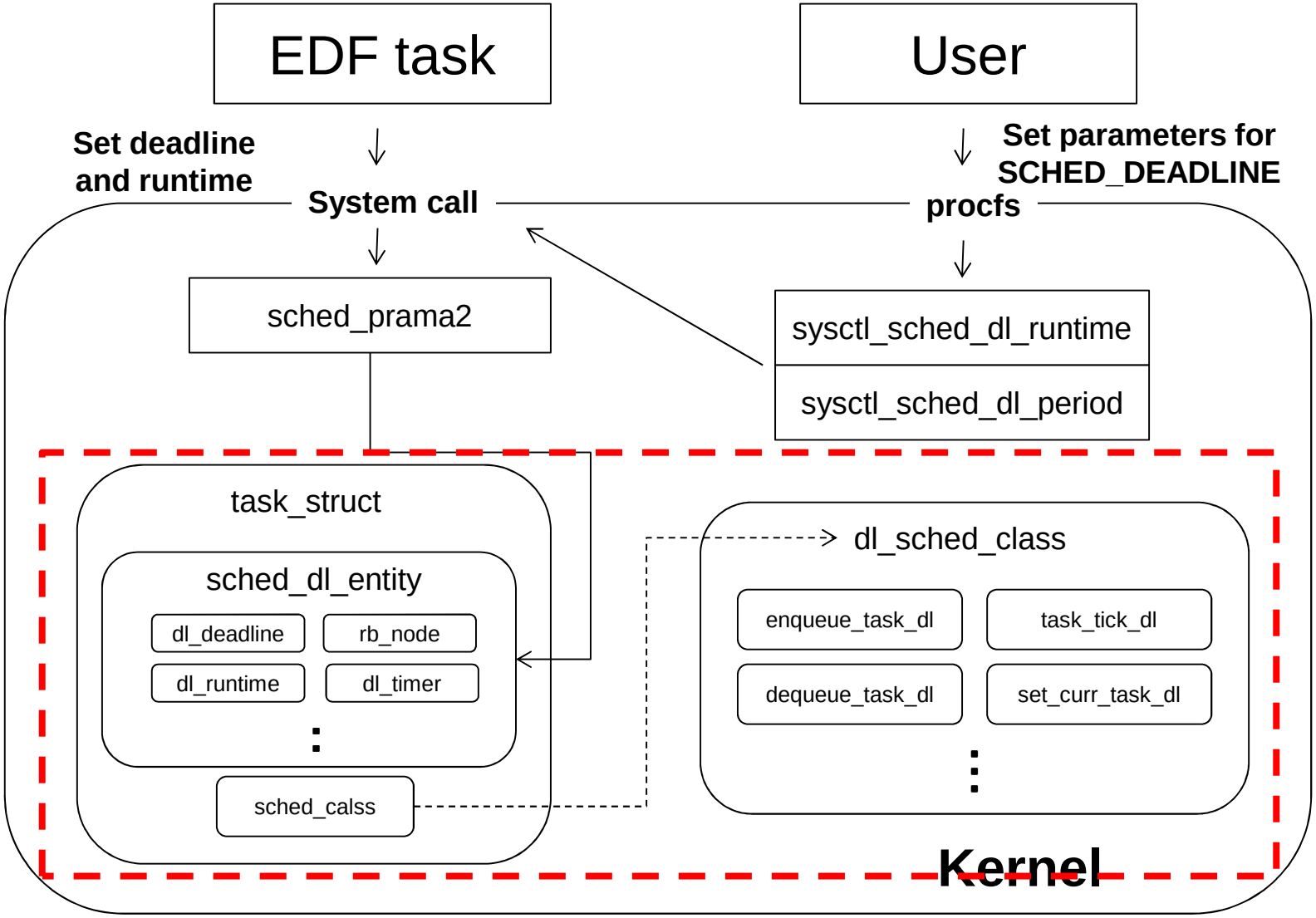
```
struct sched_param2 {
        int sched_priority;
        unsigned int sched_flags;
        u64 sched_runtime;
        u64 sched_deadline;
        u64 sched_period;

        u64 __unused[12];
};
```

Run a EDF task

- **Schedtool**
  - # schedtool -E -t 10000:100000 -a 0 -e ./yes
  - Options
    -E : a task runs on SCHED_DEADLINE
    -t : <execution time> and <period> in micro seconds
    -a : Affinity mask
    -e : command

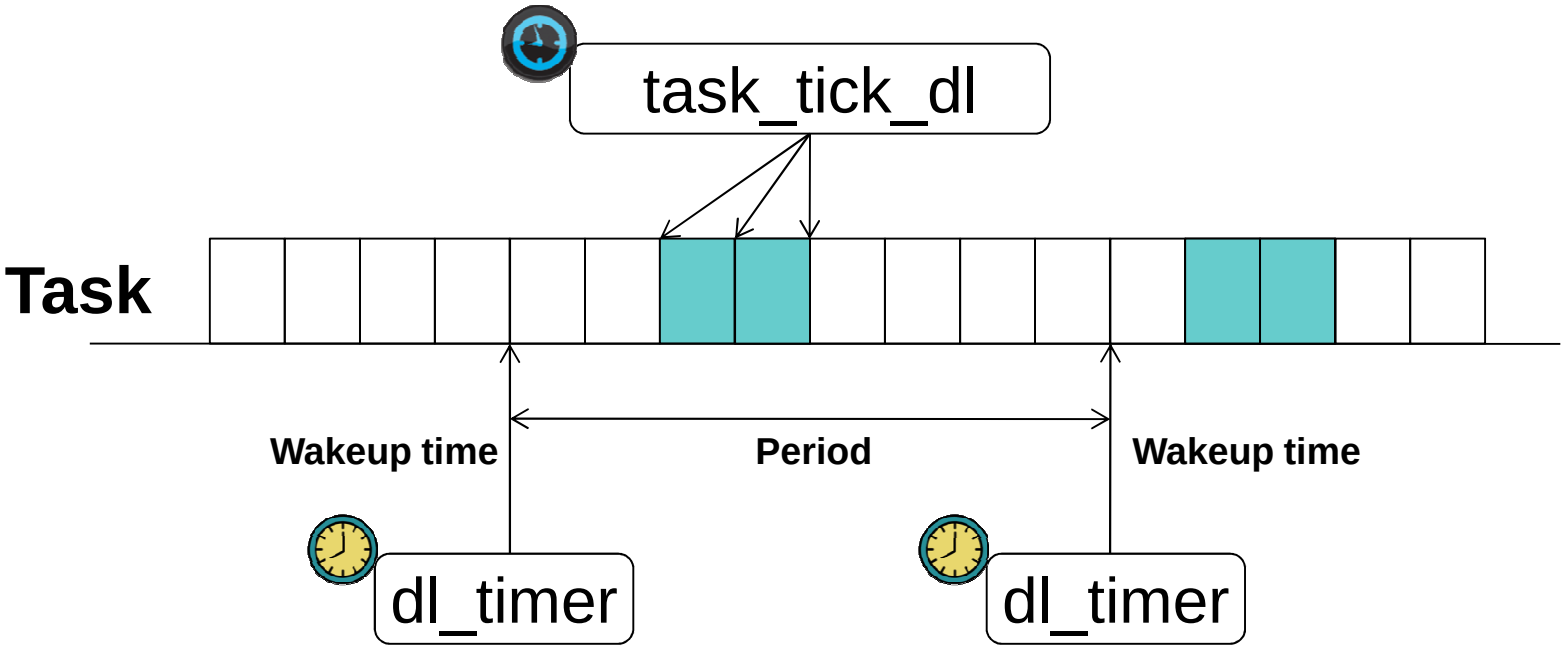
- **System call**
  - sched_setscheduler2()

Budget management for EDF tasks

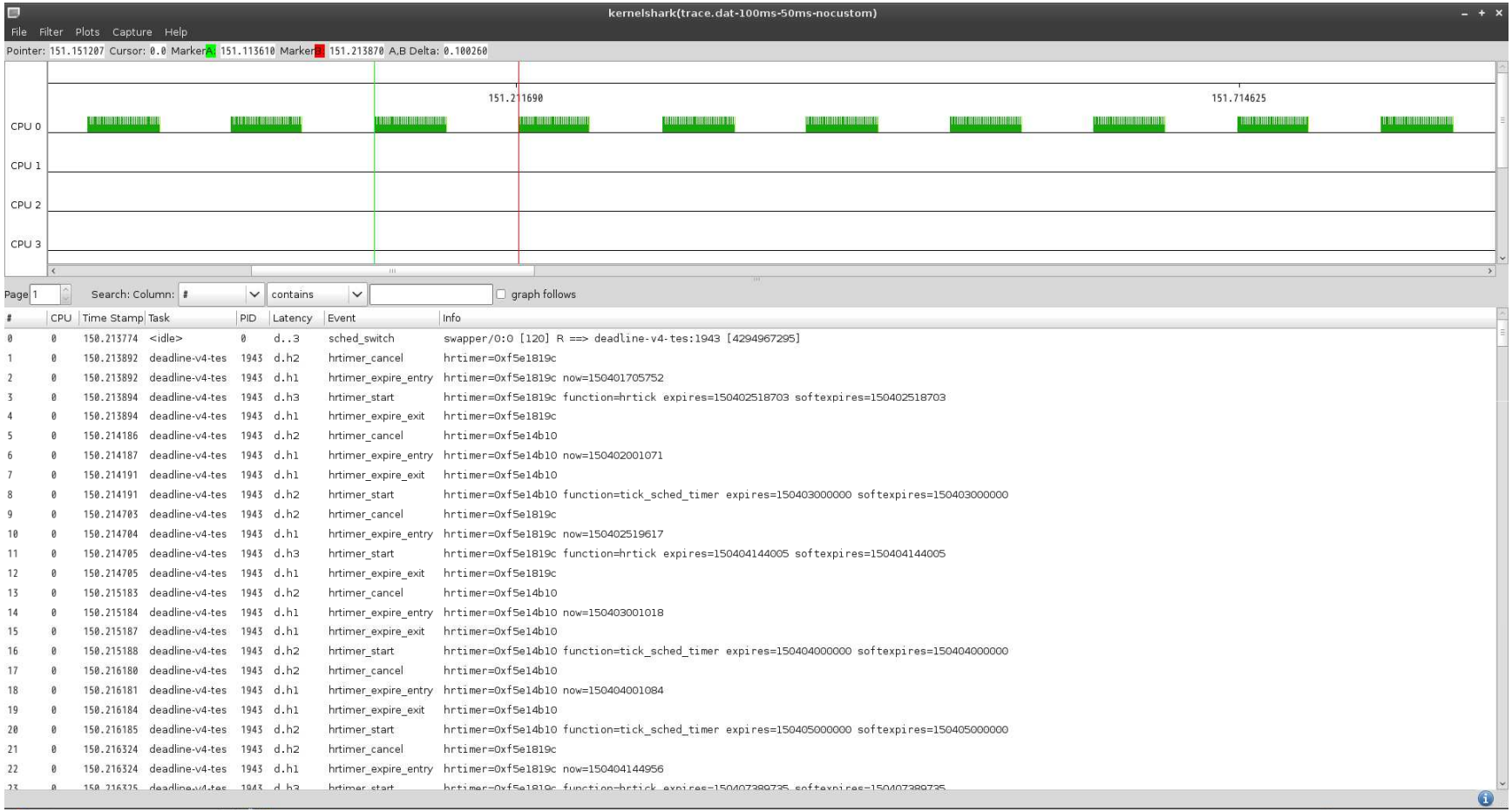


Budget management for EDF tasks

- **Each task on SCHED_DEADLINE has budget which allows it to use CPU**
- **Budget management**
  - Refill budget : dl_timer (high resolution timer)
  - Use budget : task_tick_dl (tick based)



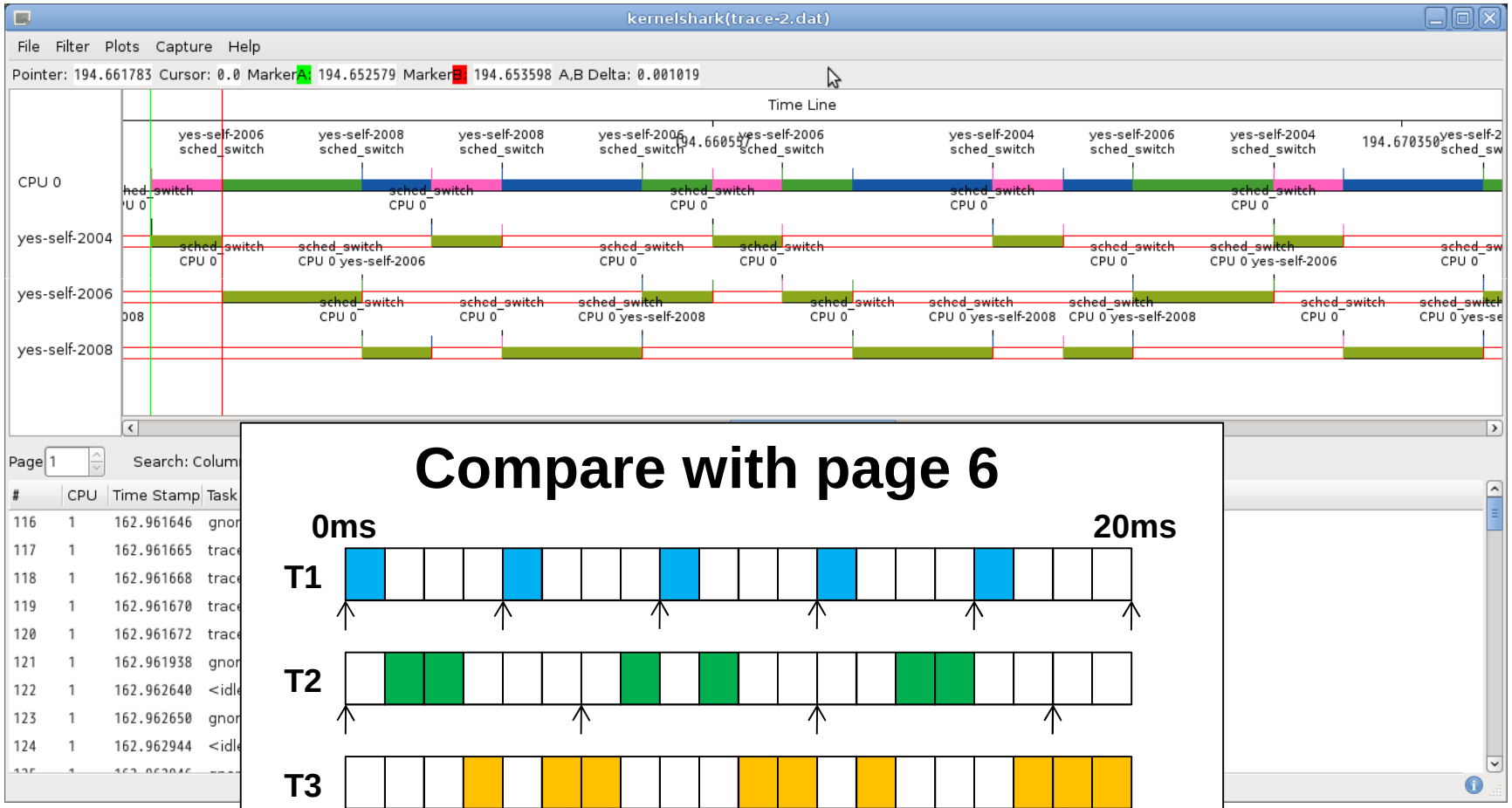
Evaluation (Period:100ms, Budget: 50ms)



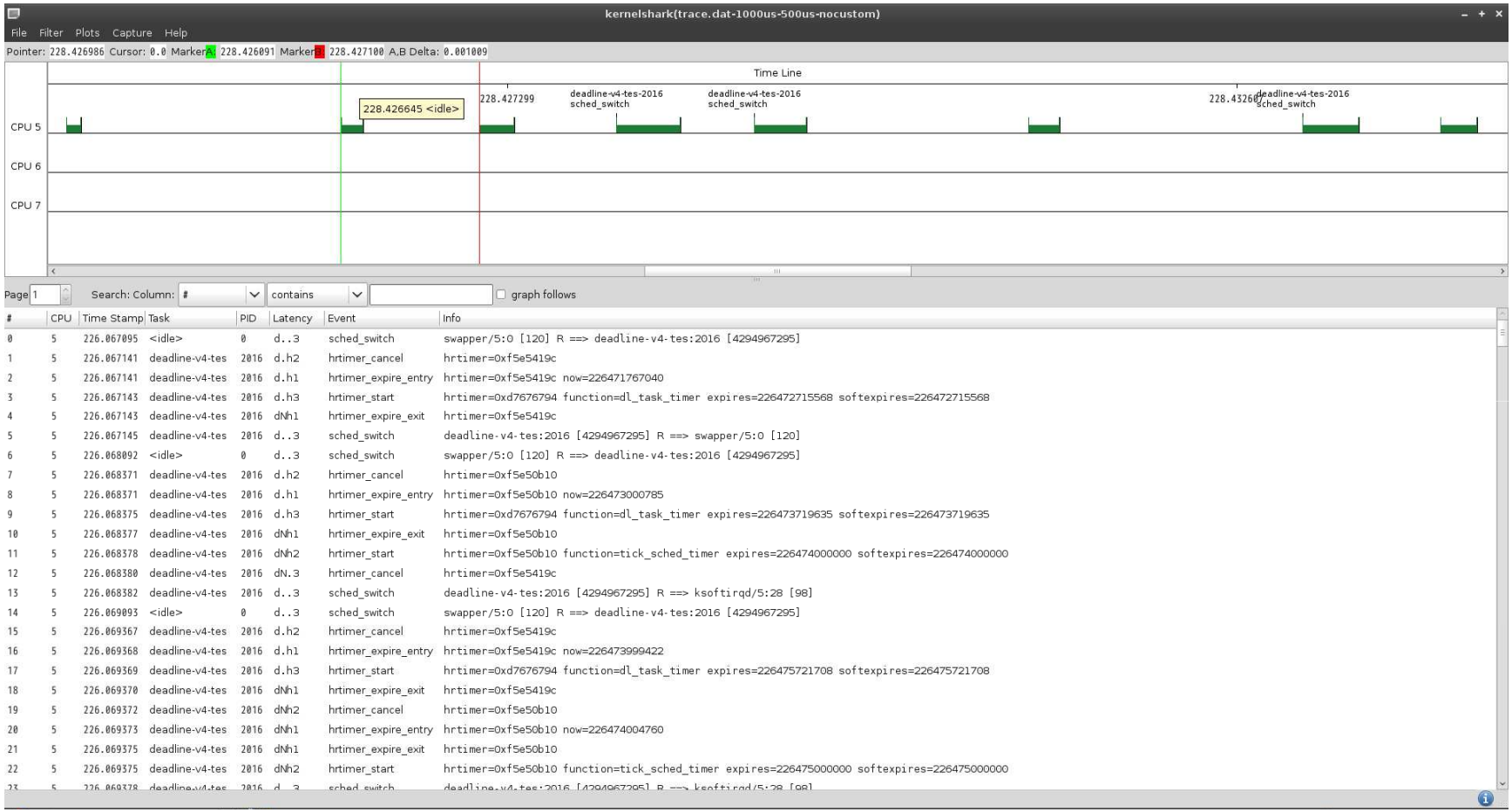
#	CPU	Time Stamp	Task	PID	Latency	Event	Info
0	0	150.213774	<idle>	0	d..3	sched_switch	swapper/0:0 [120] R ==> deadline-v4-tes:1943 [4294967295]
1	0	150.213892	deadline-v4-tes	1943	d.h2	hrtimer_cancel	hrtimer=0xf5e1819c
2	0	150.213892	deadline-v4-tes	1943	d.h1	hrtimer_expire_entry	hrtimer=0xf5e1819c now=150401705752
3	0	150.213894	deadline-v4-tes	1943	d.h3	hrtimer_start	hrtimer=0xf5e1819c function=hrtick expires=150402518703 softexpires=150402518703
4	0	150.213894	deadline-v4-tes	1943	d.h1	hrtimer_expire_exit	hrtimer=0xf5e1819c
5	0	150.214186	deadline-v4-tes	1943	d.h2	hrtimer_cancel	hrtimer=0xf5e14b10
6	0	150.214187	deadline-v4-tes	1943	d.h1	hrtimer_expire_entry	hrtimer=0xf5e14b10 now=150402001071
7	0	150.214191	deadline-v4-tes	1943	d.h1	hrtimer_expire_exit	hrtimer=0xf5e14b10
8	0	150.214191	deadline-v4-tes	1943	d.h2	hrtimer_start	hrtimer=0xf5e14b10 function=tick_sched_timer expires=150403000000 softexpires=150403000000
9	0	150.214703	deadline-v4-tes	1943	d.h2	hrtimer_cancel	hrtimer=0xf5e1819c
10	0	150.214704	deadline-v4-tes	1943	d.h1	hrtimer_expire_entry	hrtimer=0xf5e1819c now=150402519617
11	0	150.214705	deadline-v4-tes	1943	d.h3	hrtimer_start	hrtimer=0xf5e1819c function=hrtick expires=150404144005 softexpires=150404144005
12	0	150.214705	deadline-v4-tes	1943	d.h1	hrtimer_expire_exit	hrtimer=0xf5e1819c
13	0	150.215183	deadline-v4-tes	1943	d.h2	hrtimer_cancel	hrtimer=0xf5e14b10
14	0	150.215184	deadline-v4-tes	1943	d.h1	hrtimer_expire_entry	hrtimer=0xf5e14b10 now=150403001018
15	0	150.215187	deadline-v4-tes	1943	d.h1	hrtimer_expire_exit	hrtimer=0xf5e14b10
16	0	150.215188	deadline-v4-tes	1943	d.h2	hrtimer_start	hrtimer=0xf5e14b10 function=tick_sched_timer expires=150404000000 softexpires=150404000000
17	0	150.216180	deadline-v4-tes	1943	d.h2	hrtimer_cancel	hrtimer=0xf5e14b10
18	0	150.216181	deadline-v4-tes	1943	d.h1	hrtimer_expire_entry	hrtimer=0xf5e14b10 now=150404001084
19	0	150.216184	deadline-v4-tes	1943	d.h1	hrtimer_expire_exit	hrtimer=0xf5e14b10
20	0	150.216185	deadline-v4-tes	1943	d.h2	hrtimer_start	hrtimer=0xf5e14b10 function=tick_sched_timer expires=150405000000 softexpires=150405000000
21	0	150.216324	deadline-v4-tes	1943	d.h2	hrtimer_cancel	hrtimer=0xf5e1819c
22	0	150.216324	deadline-v4-tes	1943	d.h1	hrtimer_expire_entry	hrtimer=0xf5e1819c now=150404144956
23	0	150.216325	deadline-v4-tes	1943	d.h3	hrtimer_start	hrtimer=0xf5e1819c function=hrtick expires=150407389735 softexpires=150407389735

Evaluation

- **Task T1: budget 1ms period 4ms**
- **Task T2: budget 2ms period 6ms**
- **Task T3: budget 3ms period 8ms**



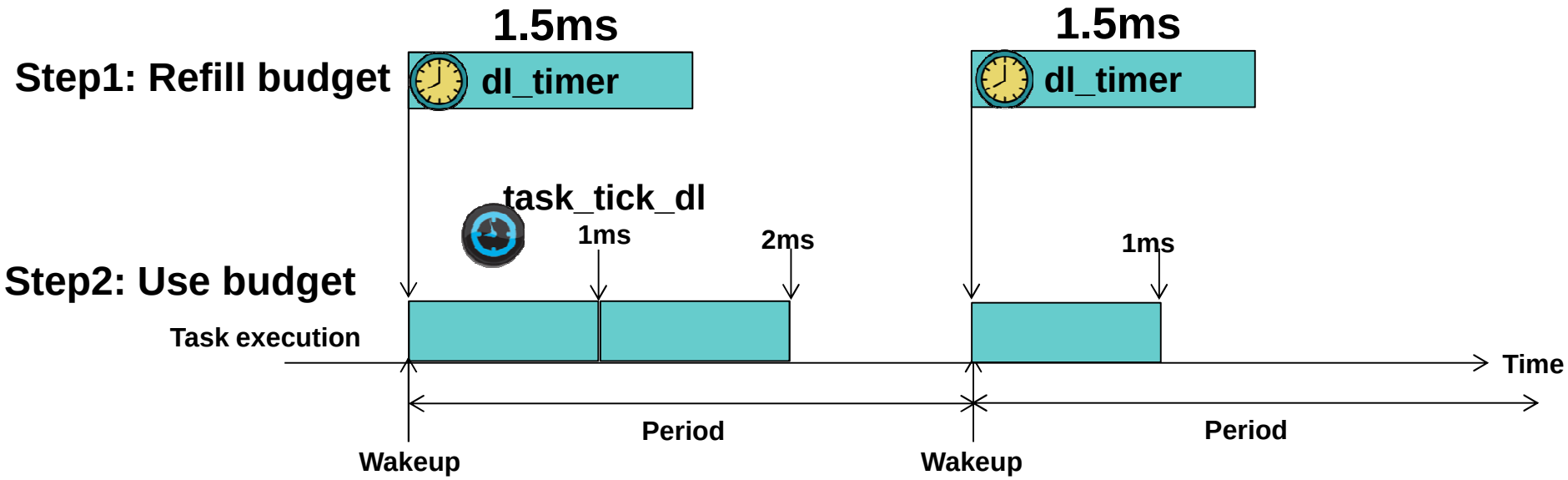
Evaluation (Period: 1ms, Budget: 0.5ms)



#	CPU	Time Stamp	Task	PID	Latency	Event	Info
0	5	226.067095	<idle>	0	d..3	sched_switch	swapper/5:0 [120] R ==> deadline-v4-tes:2016 [4294967295]
1	5	226.067141	deadline-v4-tes	2016	d.h2	hrtimer_cancel	hrtimer=0xf5e5419c
2	5	226.067141	deadline-v4-tes	2016	d.h1	hrtimer_expire_entry	hrtimer=0xf5e5419c now=226471767040
3	5	226.067143	deadline-v4-tes	2016	d.h3	hrtimer_start	hrtimer=0xd7676794 function=dl_task_timer expires=226472715568 softexpires=226472715568
4	5	226.067143	deadline-v4-tes	2016	dNh1	hrtimer_expire_exit	hrtimer=0xf5e5419c
5	5	226.067145	deadline-v4-tes	2016	d..3	sched_switch	deadline-v4-tes:2016 [4294967295] R ==> swapper/5:0 [120]
6	5	226.068092	<idle>	0	d..3	sched_switch	swapper/5:0 [120] R ==> deadline-v4-tes:2016 [4294967295]
7	5	226.068371	deadline-v4-tes	2016	d.h2	hrtimer_cancel	hrtimer=0xf5e50b10
8	5	226.068371	deadline-v4-tes	2016	d.h1	hrtimer_expire_entry	hrtimer=0xf5e50b10 now=226473000785
9	5	226.068375	deadline-v4-tes	2016	d.h3	hrtimer_start	hrtimer=0xd7676794 function=dl_task_timer expires=226473719635 softexpires=226473719635
10	5	226.068377	deadline-v4-tes	2016	dNh1	hrtimer_expire_exit	hrtimer=0xf5e50b10
11	5	226.068378	deadline-v4-tes	2016	dNh2	hrtimer_start	hrtimer=0xf5e50b10 function=tick_sched_timer expires=226474000000 softexpires=226474000000
12	5	226.068380	deadline-v4-tes	2016	dN.3	hrtimer_cancel	hrtimer=0xf5e5419c
13	5	226.068382	deadline-v4-tes	2016	d..3	sched_switch	deadline-v4-tes:2016 [4294967295] R ==> ksoftirqd/5:28 [98]
14	5	226.069093	<idle>	0	d..3	sched_switch	swapper/5:0 [120] R ==> deadline-v4-tes:2016 [4294967295]
15	5	226.069367	deadline-v4-tes	2016	d.h2	hrtimer_cancel	hrtimer=0xf5e5419c
16	5	226.069368	deadline-v4-tes	2016	d.h1	hrtimer_expire_entry	hrtimer=0xf5e5419c now=226473999422
17	5	226.069369	deadline-v4-tes	2016	d.h3	hrtimer_start	hrtimer=0xd7676794 function=dl_task_timer expires=226475721708 softexpires=226475721708
18	5	226.069370	deadline-v4-tes	2016	dNh1	hrtimer_expire_exit	hrtimer=0xf5e5419c
19	5	226.069372	deadline-v4-tes	2016	dNh2	hrtimer_cancel	hrtimer=0xf5e50b10
20	5	226.069373	deadline-v4-tes	2016	dNh1	hrtimer_expire_entry	hrtimer=0xf5e50b10 now=226474004760
21	5	226.069375	deadline-v4-tes	2016	dNh1	hrtimer_expire_exit	hrtimer=0xf5e50b10
22	5	226.069375	deadline-v4-tes	2016	dNh2	hrtimer_start	hrtimer=0xf5e50b10 function=tick_sched_timer expires=226475000000 softexpires=226475000000
23	5	226.069378	deadline-v4-tes	2016	d..3	sched_switch	deadline-v4-tes:2016 [4294967295] R ==> ksoftirqd/5:28 [98]

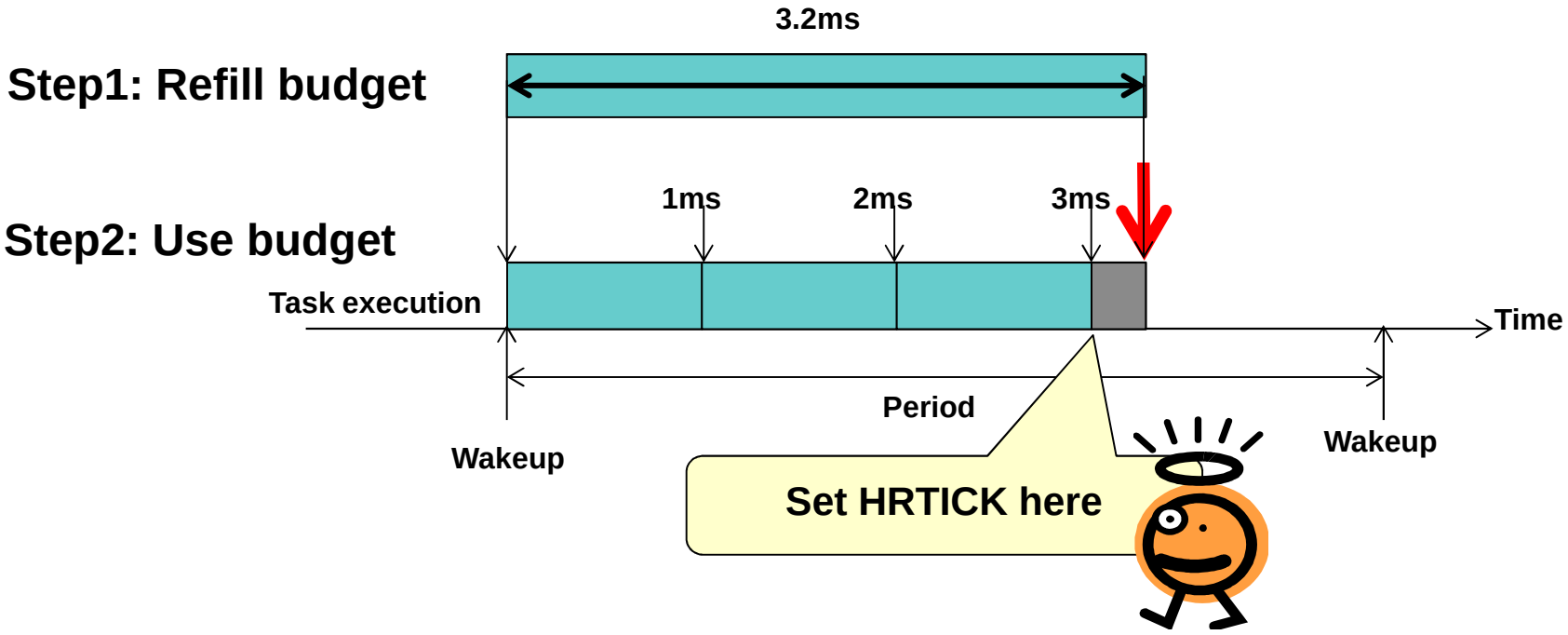
Budget management for EDF tasks

- **Each task on SCHED_DEADLINE has budget which allows it to use CPU**
- **Budget management**
  - Refill budget : dl_timer (high resolution timer)
  - Use budget : task_tick_dl (tick based)
- **An Issue**
  - Difficult to keep task's budget if the budget has micro seconds granularity

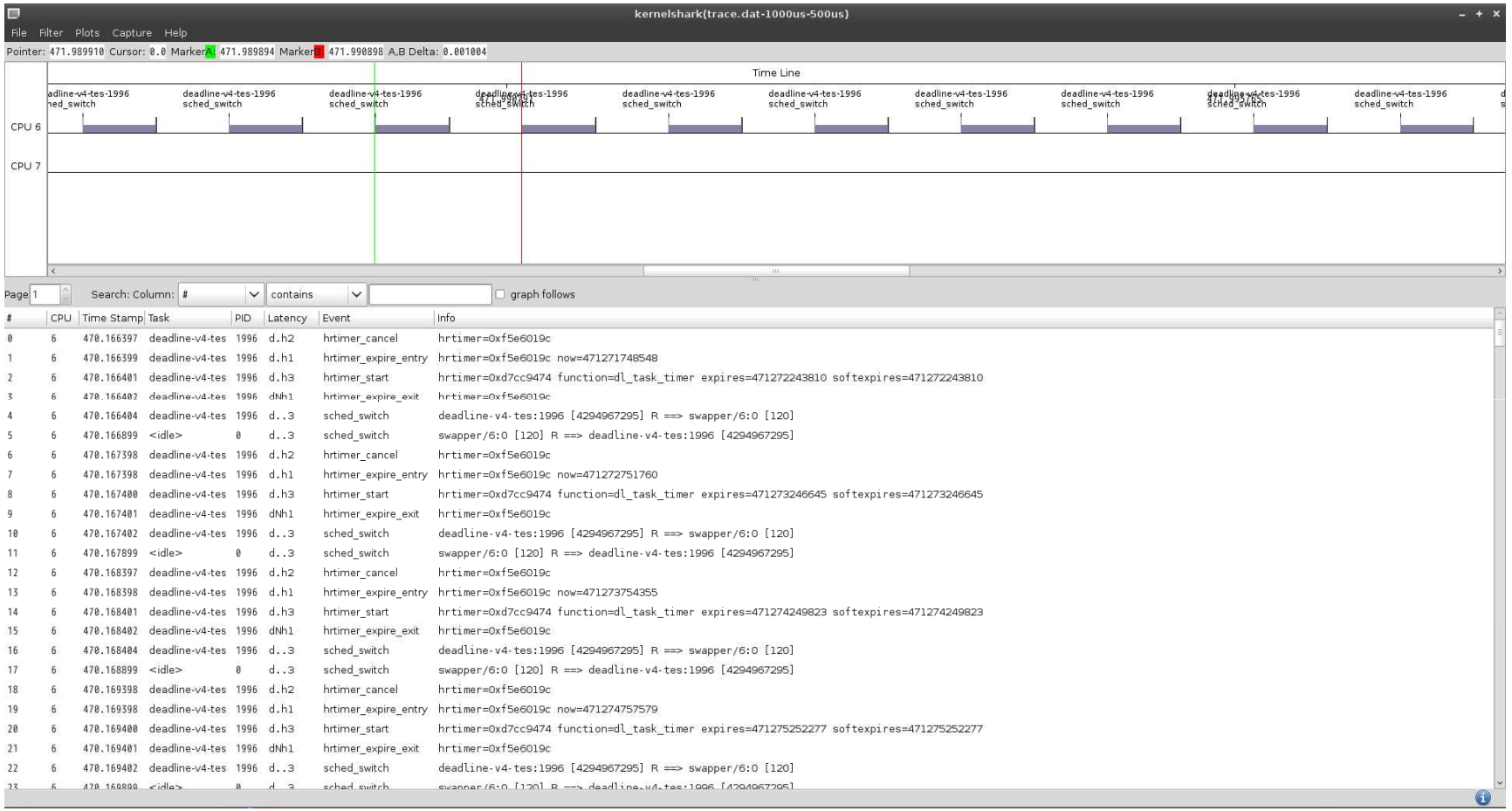


Support for micro seconds granularity

- **Overview**
  - When a task's budget is less than 1ms, set HRTICK for the rest of budget



Evaluation (Cycle: 1ms, Budget: 0.5ms)



#	CPU	Time Stamp	Task	PID	Latency	Event	Info
0	6	470.166397	deadline-v4-tes	1996	d.h2	hrtimer_cancel	hrtimer=0xf5e6019c
1	6	470.166399	deadline-v4-tes	1996	d.h1	hrtimer_expire_entry	hrtimer=0xf5e6019c now=471271748548
2	6	470.166401	deadline-v4-tes	1996	d.h3	hrtimer_start	hrtimer=0xd7cc9474 function=dl_task_timer expires=471272243810 softexpires=471272243810
3	6	470.166402	deadline-v4-tes	1996	dNh1	hrtimer_expire_exit	hrtimer=0xf5e6019c
4	6	470.166404	deadline-v4-tes	1996	d..3	sched_switch	deadline-v4-tes:1996 [4294967295] R ==> swapper/6:0 [120]
5	6	470.166899	<idle>	0	d..3	sched_switch	swapper/6:0 [120] R ==> deadline-v4-tes:1996 [4294967295]
6	6	470.167398	deadline-v4-tes	1996	d.h2	hrtimer_cancel	hrtimer=0xf5e6019c
7	6	470.167398	deadline-v4-tes	1996	d.h1	hrtimer_expire_entry	hrtimer=0xf5e6019c now=471272751760
8	6	470.167400	deadline-v4-tes	1996	d.h3	hrtimer_start	hrtimer=0xd7cc9474 function=dl_task_timer expires=471273246645 softexpires=471273246645
9	6	470.167401	deadline-v4-tes	1996	dNh1	hrtimer_expire_exit	hrtimer=0xf5e6019c
10	6	470.167402	deadline-v4-tes	1996	d..3	sched_switch	deadline-v4-tes:1996 [4294967295] R ==> swapper/6:0 [120]
11	6	470.167899	<idle>	0	d..3	sched_switch	swapper/6:0 [120] R ==> deadline-v4-tes:1996 [4294967295]
12	6	470.168397	deadline-v4-tes	1996	d.h2	hrtimer_cancel	hrtimer=0xf5e6019c
13	6	470.168398	deadline-v4-tes	1996	d.h1	hrtimer_expire_entry	hrtimer=0xf5e6019c now=471273754355
14	6	470.168401	deadline-v4-tes	1996	d.h3	hrtimer_start	hrtimer=0xd7cc9474 function=dl_task_timer expires=471274249823 softexpires=471274249823
15	6	470.168402	deadline-v4-tes	1996	dNh1	hrtimer_expire_exit	hrtimer=0xf5e6019c
16	6	470.168404	deadline-v4-tes	1996	d..3	sched_switch	deadline-v4-tes:1996 [4294967295] R ==> swapper/6:0 [120]
17	6	470.168899	<idle>	0	d..3	sched_switch	swapper/6:0 [120] R ==> deadline-v4-tes:1996 [4294967295]
18	6	470.169398	deadline-v4-tes	1996	d.h2	hrtimer_cancel	hrtimer=0xf5e6019c
19	6	470.169398	deadline-v4-tes	1996	d.h1	hrtimer_expire_entry	hrtimer=0xf5e6019c now=471274757579
20	6	470.169400	deadline-v4-tes	1996	d.h3	hrtimer_start	hrtimer=0xd7cc9474 function=dl_task_timer expires=471275252277 softexpires=471275252277
21	6	470.169401	deadline-v4-tes	1996	dNh1	hrtimer_expire_exit	hrtimer=0xf5e6019c
22	6	470.169402	deadline-v4-tes	1996	d..3	sched_switch	deadline-v4-tes:1996 [4294967295] R ==> swapper/6:0 [120]
23	6	470.169899	<idle>	0	d..3	sched_switch	swapper/6:0 [120] R ==> deadline-v4-tes:1996 [4294967295]

Advantage and Disadvantage

- **Advantage**
  - Easy to support high resolution budget
- **Disadvantage**
  - Increase overhead

Summary of scheduling granularity improvement

- **An Enhancement for budget management**
  - Support fine grained budget such as 100 micro seconds
  - HRTICK is needed to support fine grained budget

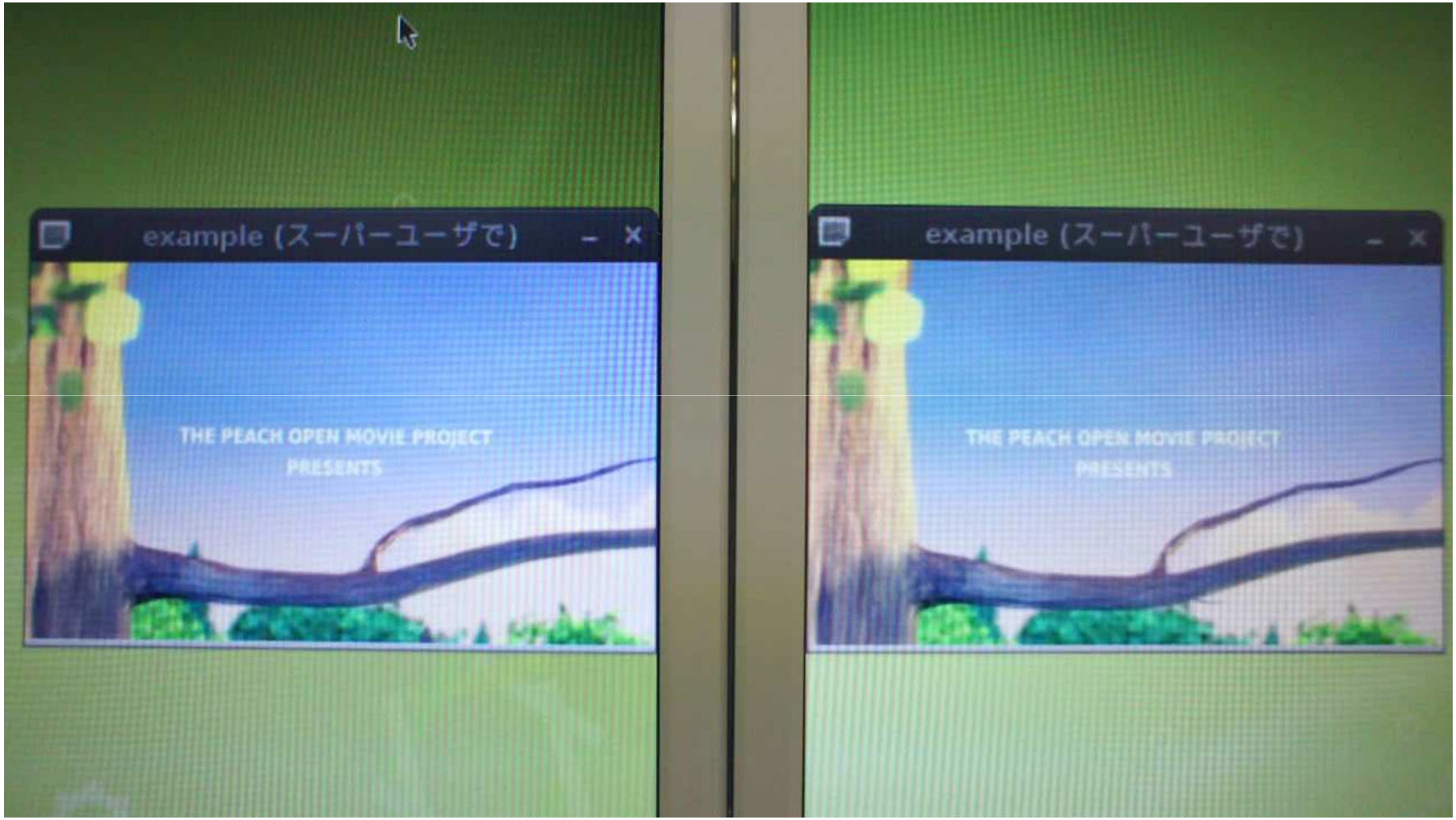
DEMO

- **“Big Buck Bunny”**
  - http://www.bigbuckbunny.org/
  - (c) copyright 2008, Blender Foundation
  - License: Creative Commons Attribution 3.0

- **Play the movie about 3 times faster than normal speed**
  - Requirement for a frame decoding
    - Period: 10ms
    - CPU budget: 3.6ms
  - When a frame decode is not finished in a period, the player shows “DEADLINE MISS” on screen and add 3ms penalty

Let’s play the movie

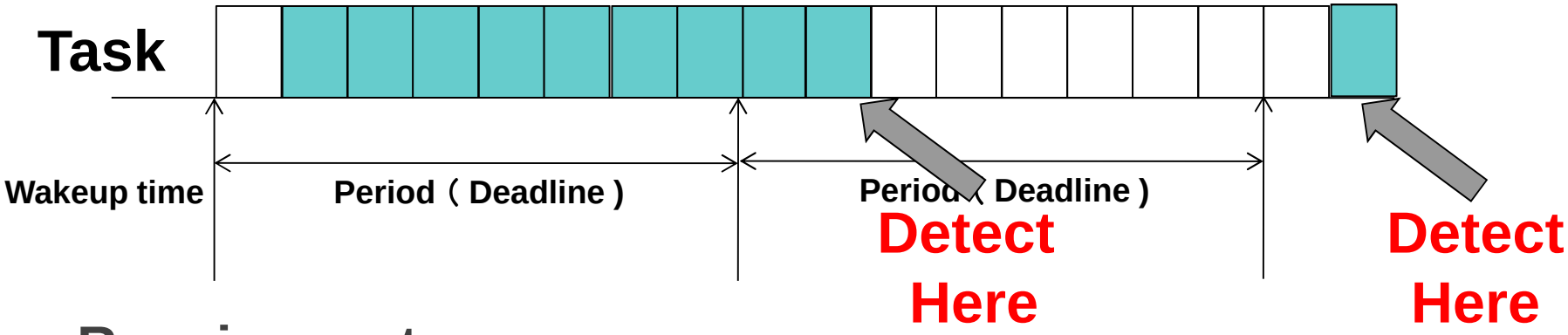
Original **Modified**



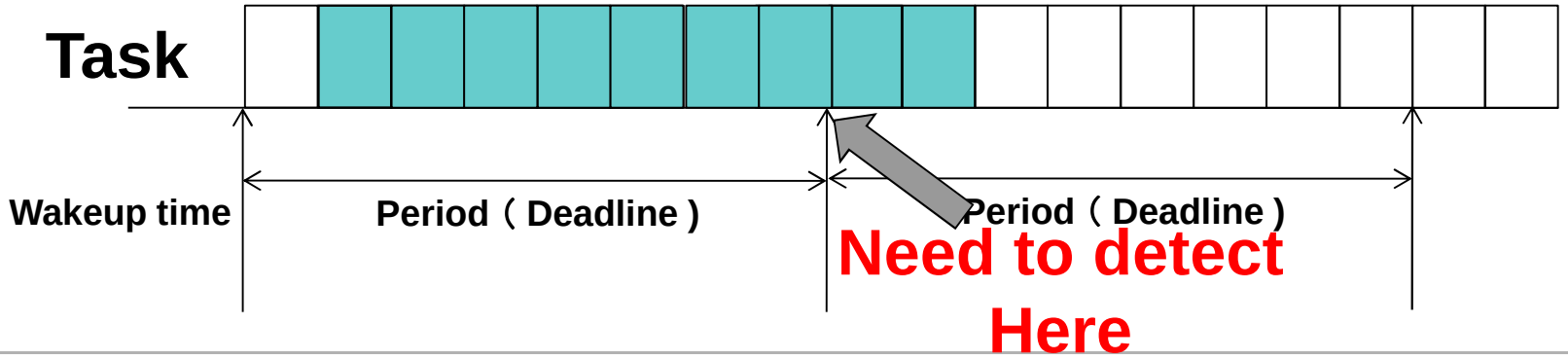
Note: Both movies are running on same spec x86 based PC

Requirement for deadline miss detection

- **Current behavior**
  - Have a counter to count the number of deadline misses
  - Not able to control tasks that missed the deadline



- **Requirement**
  - Detect the deadline miss at the begging of period



Current limitation of SCHED_DEADLINE

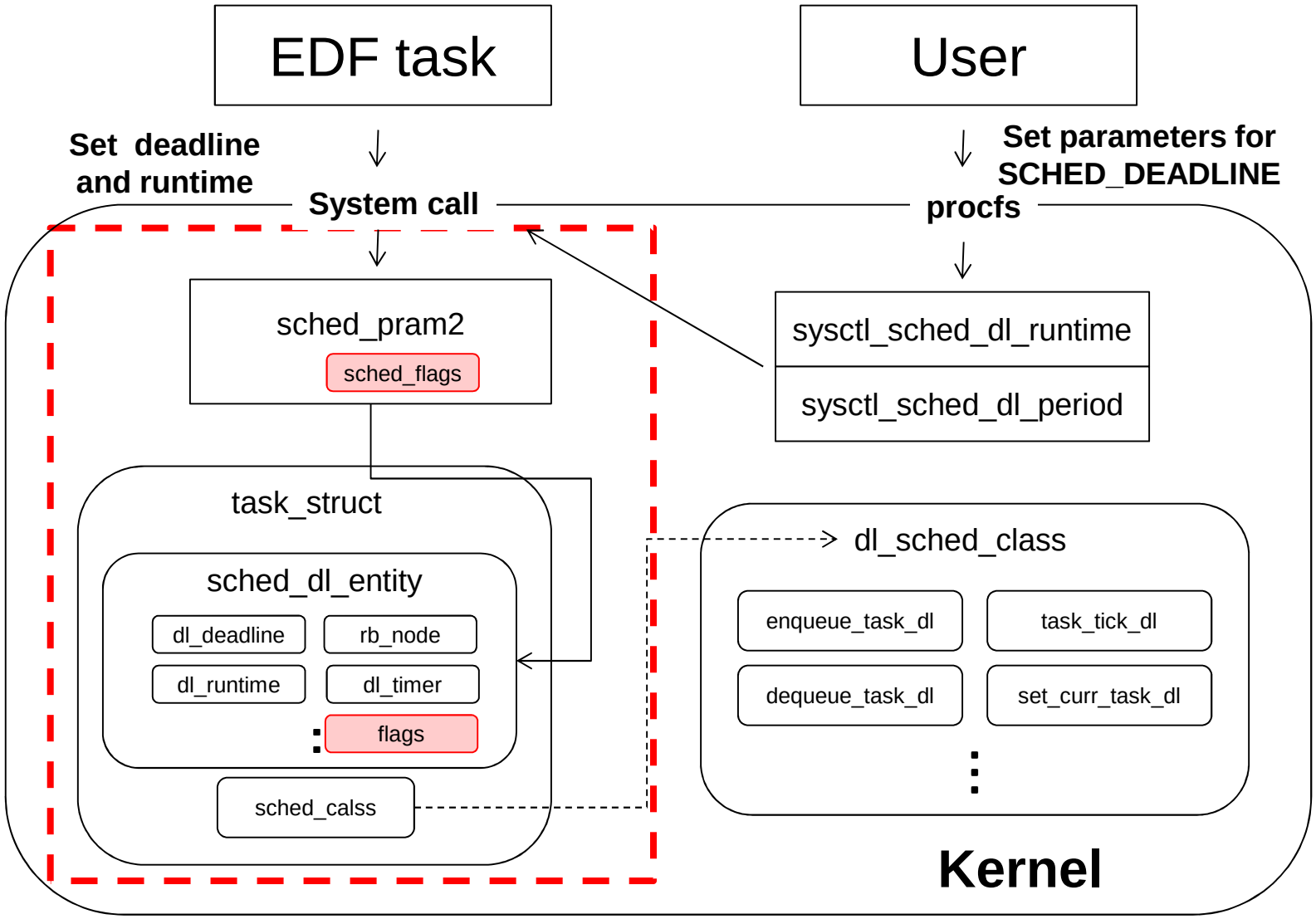
- **SCHED_DEADLINE is able to count the number of deadline misses**
- **SCHED_DEADLINE is not able to control tasks that missed the deadline**

Possible approaches to detect deadline misses

- **Userland**
  - Monitor process or thread
    - Run at the end or begging of each period
  - A pair of periodic timer and signal handler
    - Have to cancel if the process meet the deadline

      or
    - Have to catch the signal to check the deadline miss status
  - …
- **Kernel**
  - This presentation takes this approach

Overview of SCHED_DEADLINE



Overview of deadline miss detection

- **Set parameters to sched_param2.sched_flags**

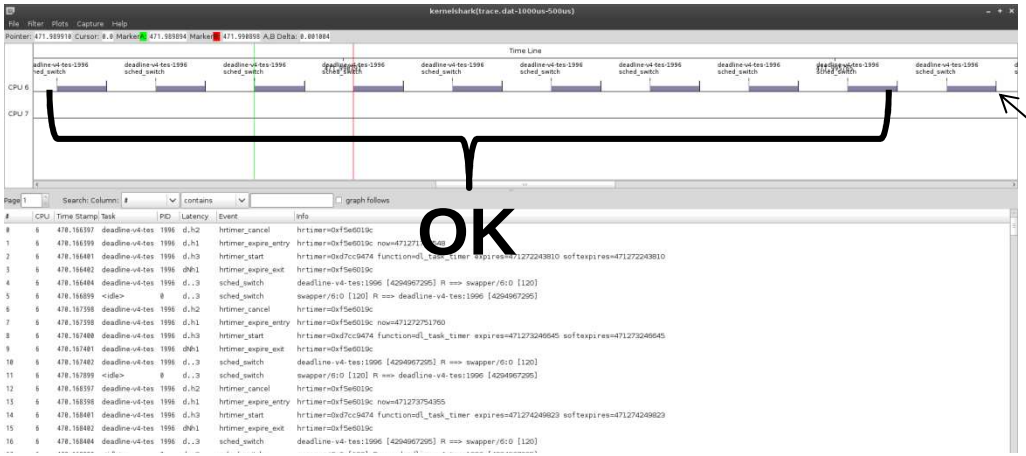
- **Parameters for sched_flags**
  - Enable or disable the deadline miss detection
    - SCHED_DL_DMISS_DETECT_DISABLE
    - SCHED_DL_DMISS_DETECT_ENABLE
  - Behaviors
    - next slide..

Behaviors (sched_param2.sched_flags)

- **Nothing to do**
  - SCHED_DL_DMISS_ACT_NONE
  - When the SCHED_DL_DMISS_DETECT_ENABLE flag is set, the kernel will count the number of deadline misses on /proc/sys/kernel/sched_dl_dmiss_count
- **Send a signal**
  - To the process itself
    - SCHED_DL_DMISS_ACT_SEND_SIG_OWN
      - Send a SIGXCPU signal
    - SCHED_DL_DMISS_ACT_PROCESS_END
      - Send a SIGKILL signal
    - SCHED_DL_DMISS_ACT_PROCESS_STOP
      - Send a SIGSTOP signal
  - To the other task
    - Set the PID to /proc/sys/kernel/sched_dl_dmiss_sig_pid
    - SCHED_DL_DMISS_ACT_SEND_SIG_OTHER_WITH_RUN
      - Send a signal to the specified process
      - Deadline missed process continues to run
    - SCHED_DL_DMISS_ACT_SEND_SIG_OTHER_WITH_STOP
      - Send a signal to the specified process
      - Deadline missed process is stoped

Evaluation

- **Behavior of process**
  - Period: 10ms
  - Budget: 5ms
    - Most of case the process is able to call sched_yield() between 4.5ms to 4.8ms
    - The process missed the deadline at the last period in the log



Conclusion

- **SCHED_DEADLINE is useful for real time systems**

- **This presentation enhanced SCHED_DEADLINE**
  - For budget management
    - Support fine grained budget such as 100 micro seconds
    - HRTICK is needed to support fine grained budget
  - For deadline miss detection
    - Add a function to control deadline missed tasks

- **All source code are available at the following URL**
  - https://github.com/ystk/sched-deadline/tree/dlmiss-detection-dev

Thank you